将棋における重い評価関数の利用効率化と非対称方向性指向に基づく評価関数の設計

濱田 剛旭[1]，橋本 剛[2]，飯田 弘之[2,3]
[1] 静岡大学情報学研究科情報学専攻 [2] 北陸先端科学技術大学院大学情報科学研究科
[3] 科学技術振興機構さきがけ研究 21

概要

将棋プログラムの開発には探索と評価関数という大きな 2 つの柱がある．大駒の自由度などの重い評価関数は差分計算を用いて容易に求めることができないため, 探索の度に呼び出すと多くの計算時間が割かれてしまう．α, β 値と十分差がある場合は重い評価関数を呼ばないことで探索速度を向上させる手法に関して, 本稿ではその適切な範囲を検証する．続いて非対称方向性指向に基づく重い評価関数の設計を提案する．評価関数の設計にあたってすべての方向を等しく評価すれば実装は容易である．しかしそれでは不十分なことが多く, 重要な一方向だけを詳しく調べることで局面評価をより正確に行うことができる．このような非対称方向性指向に基づく評価関数として敵陣にいる足の遅い駒と大駒の自由度について実装し, 評価する．

Effective usage of heavy evaluation function and a design of Directional Asymmetric Orientation evaluation function for Shogi

Takeaki Hamada[1]，Tsuyoshi Hashimoto[2], Hiroyuki Iida[2,3]
[1] Department of Computer Science, Shizuoka University
[2] School of Information Science, Japan Advanced Institute of Science and Tecnology
[3] PRESTO, Japan Science and Technology Agency

Abstract

Both search and evaluetion functions are cores of developing computer shogi. Heavy evaluation functions cannot be easily calculated by difference calculation. Therefore, calling heavy evaluation functions at every node needs a lot of time. We verify an appropriate range on a method for improving search speed by not calling heavy evaluation functions in the case of having enough difference with α or β value. Besides, we propose the design of heavy evaluation functions based on Directional Asymmetric Orientation. Implementation is easy if all directions are equally evaluated, nevertheless it is inadequate in many cases. Examining carefully only toward one direction makes the evaluation more accurate. We implement "short range pieces in enemy camp" and "mobility of major piece" as evaluation functions based on the Directional Asymmetric Orientation and investigate.

1 はじめに

将棋プログラムの開発は探索の効率化と評価関数の設計という 2 つの柱がある．評価関数に関しては山下の発表した手法 [1] が有名であり, 多くの将棋プログラムは駒の価値による評価や玉の安全度, 大駒の自由度などを点数化することで局面を評価している．駒の価値は計算が簡単なので差分計算を用いて容易に求めることができる．玉の安全度や大駒の自由度などは計算が複雑になるため駒の価値に比べると差分計算が容易ではない．この評価関数を探索する度に呼び出すのが一般的だが, 多くの計算時間が割かれてしまう．棚瀬 [2] は駒の価値を評価した時点で α 値より十分小さい場合, または β 値より十分大きい場合は大駒の自由度の計算が必要ではないとし, 省くことで探索速度を向上させた．ここで, α, β 値との差がどれほどであればそれらの計算を省いてよいのかが問題となる．そこで差の値 D を複数用意し, 検証を行った．

評価関数の設計においてどの方向に対しても対称な評価関数の設計が最も簡単である．しかしそれでは不十分なことが多く, 大事な方向を重点的に評価すべきである．だがこれまであまり言及されていない．ここでは非対称方向性指向 (Directional Asymmetric Orientation, 以下 DAO) に基づく評価関数の設計を提案し, その具体例を紹介していく．

2　重い評価関数

将棋プログラムの評価関数は以下を点数化することで局面を評価している.

1. 駒の価値 (位置と損得) — E1
2. 玉の安全度 (固さと自由度) — E2
3. 大駒の自由度 — E3
4. E1, E2, E3 では評価しきれない局面に対する特殊な評価 — E4

E1 は差分計算を用いることで比較的容易に差分計算することが可能である. 一方, E2, E3, E4 は計算が複雑になるため E1 と比較すると差分計算が容易ではない. よってここでは E3, E4 を重い評価関数と呼ぶことにする. 将棋が玉を詰めるゲームであることから E2 は重要であると考えたため重い評価関数には入れていない. 棚瀬が述べているようにα β法を用いたとき, E1 で求められた評価がα値より十分に小さい, またはβ値より十分に大きい場合は重い評価関数を呼び出して評価しても時間の無駄となることが多い. そのため, そのような局面では重い評価関数を呼び出さないことで無駄な計算時間を省くことが出来る.

2.1　実装及び検証

重い評価関数を呼び出さない条件はα値より小さい, またはβ値より大きいであることはすでに述べた通りであるが, α, β値とどれだけの差 D があれば呼び出す必要がないのかが問題となる. 重い評価関数によって加算される理論上の最大値以上α, β値と違えば重い評価関数を呼び出しても無駄となるので呼び出す必要は無い. そのため最大値を D に使用すれば最も正確な局面評価はできるが, かなりの頻度で重い評価関数を呼び出すことで計算時間が長くなってしまう. そこで我々の将棋プログラム (将棋倶楽部 24 におけるレーティング=2200) がコンピュータ将棋用問題 48 問を解く間に重い評価関数が何点を何回返すのかを調べた. その結果から一回の探索における重い評価関数の平均加点回数を出した. 結果を 100 点ごとに集計したものを図 1 に示す. 図 1 から 0〜99 点は 700 回以上加点され, 逆に-800 点以下, +800 点以上の加点はなりにくいことがわかる. よって最大値を D に使用することで重い評価関数を必要以上に呼び出すこととなり, 計算時間の無駄が生じると考えられる. このため現在は D を 750 点に設定し, 正確性の多少の減少には目をつぶり, 計算時間を短縮させている. 検証のため我々の将棋プログラムに必ず重い評価関数を呼び出すもの, D を 2800, 750 に取ったものの 3 つを実装し, 対局させる. 表 1 より正確性の多少の減少には目をつぶり, 計算時間を短縮させた結果将棋プログラムが強化されたことがわかった.

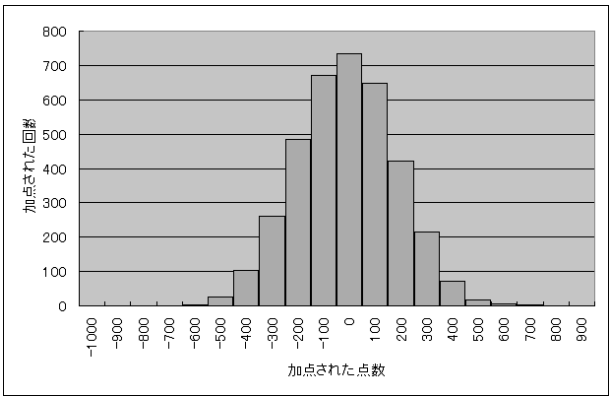

図 1: 1 回の探索における重い評価関数の平均加点回数

表 1: 自動対戦結果 1

必ず呼び出し vs $D = 750$	66 勝 133 敗 1 分
$D = 2800$ vs $D = 750$	88 勝 112 敗

3　DAO に基づく評価関数

将棋では駒の価値や玉の安全度, 大駒の自由度のみでは局面評価が上手くいかなことも多い. 探索によって補うことができる部分もあるが, 上記項目では評価するのが難しい局面では特殊な評価関数を実装して対応している. 今回は足の遅い駒 (金, 銀, 成り小駒) による相手玉の守備駒への攻めの評価関数と大駒の潜在的な相手玉への利きを用いた自由度評価の改良を実装した. これらの評価関数は一定方向のみを重視するので非対称方向性指向 (Directional Asymmetric Orientation) に基づく評価関数と呼ぶことにする.

3.1　足の遅い駒による守備駒への攻め

将棋は相手玉を詰めるゲームであるため, 相手玉やその守備駒を攻めていくのが基本である. 図 2 において 7 三のと金は相手玉のある右側へ攻めていきその守備駒と交換できれば大いに役立ったと言える. しかし 9 二の金や 9 一の香車を取れば駒

得であるため, 左側へ攻めていく手を指すことがある．この手は相手玉から離れていくため守備駒への攻めには劣る．このような事態は駒の損得重視のコンピュータ将棋の悪い癖である．そこで足の遅い駒が相手玉とは逆の方向にある駒を狙うことで相手玉から離れることを防ぐ評価関数を実装する．この評価関数は第 2 章の E4 に該当し, 対象となる駒と周辺の駒との位置から求めることができる．その計算には時間がかかるが重要な評価要素であり, プログラムの強化に繋がると考えられる．

3.1.1 足の遅い駒の評価関数の実装

以下の評価関数を我々の将棋プログラムに実装した．なお評価値の基本として歩の基本価値を 100 点としている．足の遅い駒, 相手玉, 守備駒のある X 座標をそれぞれ X, Xk, Xd とし, 足の遅い駒と守備駒の距離を $\Delta\ d$ としたとき

$$(X - Xk)(X - Xd) > 0 \text{ かつ} \Delta\ d \leqq 2$$

ならば 100 点加点するようにした．これにより足の遅い駒が守備駒に近づきやすくなると考えられる．

3.1.2 評価

金銀成り小駒に対してこの評価関数を実装し (DAO-1), 実装前のもの (DAO-0) と自動対戦させた．表 2 に結果を示す．勝ち越すことができなかったのは金銀に関してはこれまでの評価関数において十分に大きく評価されているため, 更なる加点はあまり意味を持たないのではないかと考えられる．そこで新たに成り小駒に関してのみ実装したものを用意し (DAO-2), DAO-0 と自動対戦させた。その結果それほど大きくはないものの勝ち越すことができた．さらに探索の末端付近での図 2 の局面において☗6三と を選ぶなど成り駒の活用が見られた．DAO に基づく評価関数によってプログラムが強化されたと考えられる．

表 2: 自動対戦結果 2

DAO-1vsDAO-0	100 勝 100 敗
DAO-2vsDAO-0	106 勝 94 敗

3.2 大駒の評価関数

大駒の自由度は金沢 [3] の述べているように移動できるマスを点数化することで求められる．金沢はさらに相手玉の付近などの重みを加えている．しかし移動できるマスを見ているだけでは不十分な場合も多い．図 3 の局面において, 2 八の飛車は 2 四の歩によって利きが塞がれているが, そのまま利きを伸ばしていくと相手玉へあたる．よってこの飛車は相手玉への潜在的な利きを持っており, その働きは非常に大きい．また 5 五の角は 3 三の銀によって利きが塞がれているが, この銀はピンになっているためこの角の働きも非常に大きい．このような大駒の働きは動ける範囲を点数化しただけでは正しく評価できない．そこで敵陣以外にある大駒について上方向のみ利きを伸ばしつづけることで相手玉への潜在的な利きが存在するかを判定し, 利きが存在すれば加点することにする．

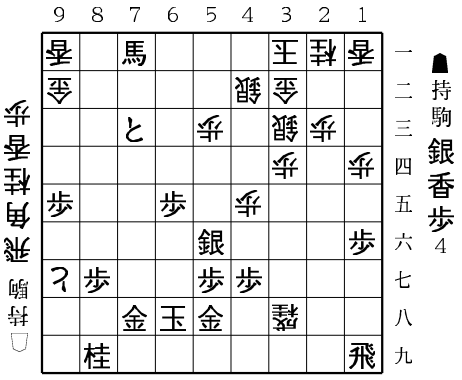

図 2: 局面例 1

3.2.1 飛車, 角の評価関数

飛車の上方向への利きを見ていき, 以下の条件のとき飛車に加点することで相手玉への利きを評価する．なお全て先手側からの視点であるため後手側は上下が逆となる．

1. 飛車の利きを止めている駒が歩で 3, 4 段目にあり, 1 段先に駒がなく 2 段先に相手玉がある (図 3)

2. 飛車の利きを止めている駒が角, 金銀成り小駒 (相手玉との距離が 2 段以内), 桂馬 (相手玉との距離が 4 段以内) であり, その先に相手玉がある (図 4)

3. 飛車の利きを止めている駒が角金銀成り小駒で 2,3,4 段目にあり, 相手玉が相手陣かつ飛車と左右同じ側にある (図 5)

以下は我々の将棋プログラムへの実装を示す．評価値の基本として歩の基本価値を 100 点としている．1. では飛車に 300 点加点しているが, 相手玉が

飛車の両隣の筋にあっても飛車の利きがあると考え 100 点加点している. ただし相手玉が 1, 9 筋にある場合は同じ筋と等しい価値があると考え 300 点加点している.

2. では利きを塞いでいる駒と飛車との連携を考慮する. そのため利きを塞いでいる駒のある場所によって加点するかどうかを判定し, 300 点加点する.

3. は 2. で加点されなかった場合のみ判定される. 2, 3 段目と 4 段目では前者がより良いため, 2, 3 段目ならば 300 点, 4 段目ならば 150 点加点している.

角の評価関数も上方向への利きを伸ばしていく. 角の利きを塞いでいる駒が相手の駒であり, その駒がピンになっていれば角に 100 点加点する (図 3).

3.2.2 評価

我々の将棋プログラムに飛車についての評価関数を実装したもの (DAO-3) と角について実装したもの (DAO-4) を用意し実装前のもの (DAO-0) とそれぞれ対戦させた. 結果を表 3 に示す. 結果より飛車, 角ともに勝ち越すことができた. それほど大きく勝ち越すことはできなかったが探索の末端付近での図 6 の局面において☗2四歩 とするなど大駒の活用が見られた. DAO に基づく大駒自由度の評価関数が有効であると言える.

表 3: 自動対戦結果 3

DAO-3vsDAO-0	104 勝 95 敗 1 分
DAO-4vsDAO-0	104 勝 96 敗

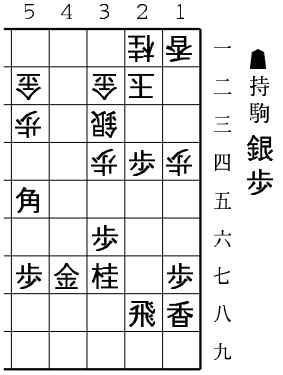

図 3: 局面例 2

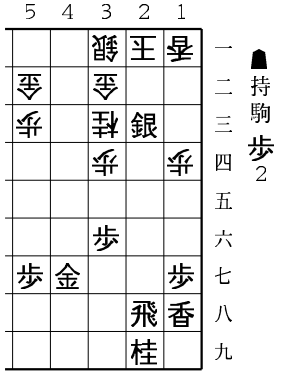

図 4: 局面例 3

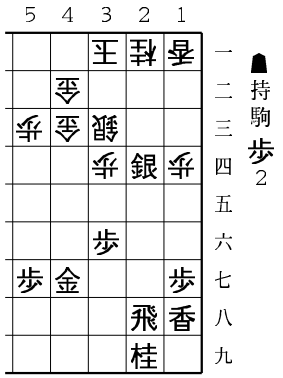

図 5: 局面例 4

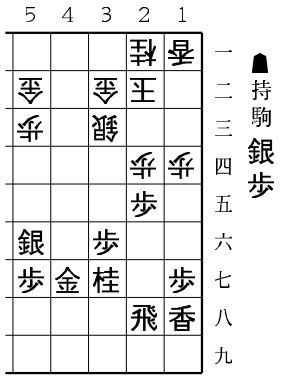

図 6: 局面例 5

4 まとめ

本稿では重い評価関数の利用効率化と DAO に基づく評価関数について実装, 検証を行った. 局面評価の正確さを下げてでも重い評価関数を呼び出す回数を減らし, 計算時間を短縮することで将棋プログラムを強化することができた. そのためには, 呼び出す条件として重い評価関数によって得られる評価の最大値よりも小さい値を α, β 値との差に使用すればよいことがわかった. DAO に基づく評価関数については足の遅い駒による守備駒への攻めと大駒の自由度に関して実装した. 成り小駒を守備駒へ近づけることで活用し, 大駒の相手玉への利きをより詳細に調べることで自由度を高めることに成功した. しかし大きく勝ち越すことができなかった. その要因として計算時間の増大が考えられ, より計算効率のよい評価関数を設計する必要があり今後の課題である.

参考文献

[1] 山下 宏, YSS-そのデータ構造, およびアルゴリズムについて. 松原 仁 編, コンピュータ将棋の進歩 2, 6 章, p112-142. 共立出版 1998.

[2] 棚瀬 寧, IS 将棋のアルゴリズム. 松原 仁 編, コンピュータ将棋の進歩 3, 1 章, p1-14. 共立出版 2000.

[3] 金沢伸一郎, 金沢将棋のアルゴリズム. 松原 仁 編, コンピュータ将棋の進歩 3, 2 章, p15-26, 共立出版 2000.